binho

应用笔记

使用 Binho Supernova 读取 DDR5 SPD Hub 并通过 DIMM 边带驱动 Corsair RGB

AN0016

文档信息

关键词	AN0016, Supernova, DDR5, SPD, SPD5118, SPD Hub, JESD300-5, 边带, I2C, I3C, SETAASA, RSTDAA, PMIC, 温度传感器, Corsair, Vengeance, RGB, CRC-8
摘要	DDR5 模块在其边带总线上带有一个 SPD5118 Hub：包含 SPD NVM、一个温度传感器，以及通往模块上其他器件的网关。 本应用笔记使用 Binho Supernova 在 I2C 模式和 I3C 模式下读取该 Hub，并驱动位于 Hub 后方的 Corsair DDR5 模块的 RGB 灯光控制器。文中给出寄存器访问序列、在一个测试台上返回正确数据的 I3C 速率、 一个在 I3C 寄存器写入中出现并使总线挂起直到模块重新上电的故障，以及两个基于 pycosmicsdk 的 Python 脚本。

Binho Inc.
42 Dore Street, Unit A, San Francisco, CA 94103

修订版 1.0
6 NOVEMBER 2026

> **测试范围**。本文档中的每个数字都来自 2026 年 10 月 6 日的一次测试台会话， 使用一个模块。随附脚本由测试台上使用的脚本整合而成，随后在同一测试台上运行（第 12 节）。

1 简介

DDR5 DIMM 提供一条独立于内存总线的两线边带（HSCL、HSDA）。在该总线上首先应答的是 SPD5118 Hub；它保存模块的 SPD 数据、报告温度，并把流量转发到模块本地总线上的其他器件，例如 PMIC，以及在带灯光的模块上的 RGB 控制器。Hub 以 I2C 模式启动，可以切换到 I3C。

本笔记介绍如何在没有主板的情况下，从主机适配器读取 Hub 及上述器件。

1.1 范围

本文档涵盖：

- 识别 SPD5118 Hub、读取其温度传感器，以及读取 1 KB 的 SPD NVM
- 读取 PMIC 状态，仅读取
- 使用 SETAASA 将 Hub 切换到 I3C、以 SDR 读取，并使用 RSTDAA 返回 I2C
- 在本测试台上返回正确数据的 I3C 速率，以及未返回正确数据的那个速率
- 位于 Hub 后方的 Corsair RGB 控制器：使其退出固件更新模式、逐个 LED 设置颜色，以及启动硬件效果

本文档不涵盖：写入 SPD NVM、通过 I3C 写入 Hub 寄存器（第 5.3 节说明原因）、来自 Hub 的 带内中断（IBI）（已尝试，未收到）、PEC，以及在 Hub 处于 I3C 模式时访问 RGB 控制器（未尝试）。

1.2 适用器件

表 1. 适用器件

角色	器件	标识	依据
主机适配器	Binho Supernova	硬件版本 C，固件 4.6.1	实测，本文档
SPD Hub	下述模块上的 SPD5118	MR0-1 `51 18`，MR2 版本 `0x14`	实测，本文档
PMIC	下述模块上的 PMIC	地址 `0x48`	只读状态，实测
RGB 控制器	Corsair Vengeance RGB RS DDR5，6 个 LED	VID `0x1B1C`；PID 在固件更新模式下为 `0x0B00`，在正常模式下为 `0x0B01`	实测，本文档
模块	Corsair `CMG16GX5M2E6000Z36`（套装中的一条 DIMM）	SPD 字节 2 `0x12`（DDR5），字节 3 `0x02`（UDIMM）	实测，本文档
其他 SPD5118 模块	任意 DDR5 UDIMM 或 SODIMM	MR0-1 `51 18`	第 5.1 和 5.2 节遵循 JESD300-5；除一个模块外未经测试
其他 Corsair DDR5 RGB 模块		其他 PID	未测试

1.3 相关文档

表 2. 相关文档

文档	标题
JESD300-5	SPD5118 Hub and Serial Presence Detect Device Standard（JEDEC）
JESD400-5	DDR5 Serial Presence Detect (SPD) Contents（JEDEC）
MIPI I3C Basic	Improved Inter Integrated Circuit Basic 规范
OpenRGB	Corsair DRAM 控制器源代码，第 5.4 节中 RGB 寄存器映射的来源

2 系统概述

表 3. 边带上看到的器件，Hub 处于 I2C 模式，100 kHz

地址	器件	应答方式
`0x50`	SPD5118 Hub	寄存器指针 + 读取；裸读取也应答
`0x48`	PMIC	寄存器指针 + 读取
`0x18`	Corsair RGB 控制器	**仅限寄存器指针 + 读取**；裸读取不被应答

该模块的 HSA 被配置为 0，这使 Hub 位于 `0x50`。Hub 把发往模块本地总线上器件的传输转发到 主机侧；JESD300-5 将这两个地址字段称为本地 ID（local ID）和主机 ID（host ID）。在 I2C 模式下，这种转发不需要任何配置。PMIC 和 RGB 控制器都通过它访问。

以下两个事实决定了下文的全部内容：

- **裸读取扫描找不到 RGB 控制器。** 用一字节读取扫描 `0x08` 到 `0x77`，只返回 `0x48` 和 `0x50`。先写入寄存器指针再读取，才能在 `0x18` 找到它。
- **在本测试台上，Hub 是一个不会被 ENTDAA 发现的 I3C 目标器件。** 收到 SETAASA 时，它将其静态地址作为动态地址；收到 RSTDAA 时返回 I2C。

3 硬件设置

3.1 所需设备

表 4. 所需设备

项目	说明
Binho Supernova	此处使用固件 4.6.1
DDR5 模块	第 1.2 节中的模块
DIMM 转接板	引出 HSCL、HSDA 和 GND，配置 HSA，并从其自身的 5 V 输入提供 VIN_BULK
转接板的 5 V 电源	此处使用转接板上的 USB-C
4 线电缆	连接 Supernova I3C LV 连接器与转接板

3.2 连接

表 5. Supernova 与 DIMM 转接板之间的连接

Supernova I3C LV 连接器	DIMM 转接板	必需
SCL	HSCL	是
SDA	HSDA	是
GND	GND	是

3.3 总线注意事项

总线运行在 **1.1 V**（设定 1100 mV，回读 1106 mV）。Supernova 仅根据电压选择 I3C 连接器：

表 6. Supernova I3C 连接器选择

请求电压	连接器
800 到 1199 mV	LV
1200 到 3300 mV	HV

> **注意：** 请求 1200 mV 会把总线切换到 HV 连接器。模块接在 LV 连接器上时，将没有任何 器件应答。请请求 1100 mV。
>
> **注意：** 模块需要在 VIN_BULK 上提供 5 V。该电压由转接板提供，适配器不提供。请先给模块 上电，再启动总线。要清除第 5.3 节中的挂起，请给模块重新上电，而不是适配器。

4 软件设置

要求：Python 3 和 pycosmicsdk（最新版本）。

```
pip install --upgrade pycosmicsdk
python ddr5_spd.py --help
python ddr5_rgb.py --help
```

两个脚本只使用 pycosmicsdk 的公开调用：`Device.open`、`I3cController.set_voltage`、`bring_up`、`legacy_i2c_read`、`legacy_i2c_write`、`sdr_read`、`ccc.setaasa` 和 `ccc.rstdaa`。两个脚本都接受 `--serial` 用于选择适配器，以及 `--voltage-mv`（默认 1100）。测试台使用的是 Ubuntu 22.04 主机，pycosmicsdk 1.6.0 和 Python 3.10。

5 寄存器与协议

5.1 I2C 模式下的 SPD5118 Hub

所有访问都是一个一字节寄存器指针，后跟一次读取。

表 7. 使用的 Hub 寄存器

寄存器	本模块上的读数	含义
MR0-1	`51 18`	器件类型，SPD5118
MR2	`0x14`	器件版本
MR5	`0x03`	能力：Hub 和温度传感器
MR11	`0x00`	I2C 传统模式：位 2:0 选择 NVM 页，位 3 选择 2 字节寻址
MR12-13	`FF 7F`	NVM 写保护，每个 64 字节块一位
MR18	I2C 下为 `0x00`，I3C 下为 `0x20`	位 5 = I3C 模式
MR49-50	两个字节	温度

温度。 MR50 位 4:0 和 MR49 位 7:0 组成一个 13 位字段，其中位 1:0 未使用。其余 11 位为 二进制补码，每 LSB 为 0.25 °C：

```
raw = (((mr50 & 0x1F) << 8) | mr49) >> 2
if raw & 0x400:
    raw -= 0x800
celsius = raw * 0.25
```

在测试台上，该模块读数为 27 到 29 °C。其限值寄存器读数为 55 °C（高）和 85 °C（临界）。

NVM。 1024 字节的 NVM 由八个 128 字节的页组成。在 1 字节寻址下，位 7 置位的指针寻址 NVM，位 6:0 为页内偏移，MR11 位 2:0 为页号。读取全部内容的方法：保存 MR11，依次把页 0 到 7 写入其中，每次从指针 `0x80` 读取 128 字节，然后写回保存的值。MR11 是唯一被写入的寄存器。 该模块的转储解码为字节 2 = `0x12`（DDR5）和字节 3 = `0x02`（UDIMM），且字节 0 到 509 上的 CRC-16（多项式 `0x1021`，初始值 0）与字节 510 和 511 中存储的值一致。

PMIC。 寄存器 R32 读数为 `0x80`（稳压器已使能），且未置位任何故障位。PMIC 仅被读取。

5.2 I3C 模式下的 SPD5118 Hub

1. 以 I3C 模式启动控制器。
2. 广播 SETAASA，器件表中包含 `0x50`。MR18 从 `0x00` 变为 `0x20`。
3. 使用寄存器指针以 SDR 读取：`sdr_read(0x50, n, subaddress=bytes([reg]))`。

4. 广播 RSTDAA。通过 I2C 读取时，MR18 再次为 `0x00`。

`init_bus()` 使用 ENTDAA 发现目标器件，在本测试台上未能启动 Hub。请使用 SETAASA。

表 8. 1.1 V 下从 Hub 进行的 I3C SDR 读取，按推挽速率

推挽速率	结果
1 MHz	20/20
5 MHz	50/50（首次运行为 20/20）
6.25 MHz	50/50
7.5 MHz	50/50
10 MHz	50/50
12.5 MHz	9/50 正确；其余 41 次在成功状态下返回了错误数据

12.5 MHz 下的失败并未被报告为错误：读取已完成，但数据错误。转接板布线在 1.1 V 下可能是 限制因素。**在较长的边带布线上请使用 4 MHz 或更低。** 已测量 1 MHz（20/20）以及 5 到 10 MHz； 两者之间的速率未测量。`ddr5_spd.py --i3c` 默认使用 1 MHz。

5.3 通过 I3C 写入寄存器

> **注意：** 在 I3C SDR 模式下对 Hub 的寄存器写入未按写入的值生效，随后总线挂起。写入为 26.5 °C 的温度限值回读为 48 °C；向 MR27 写入的 `0x01` 回读为 `0x10`。此后，每个 I3C 操作都 返回 `FW_I3C_ENGINE_BUSY`。重启 Supernova 无法清除该状态，对 LV 连接器重新上电也不行。只有 给模块重新上电才能清除。原因（Hub 的 I3C 写入格式或主机侧）尚未确定。在 I3C 模式下只进行 读取。如果需要写入，请在 I2C 模式下进行，或先写入一个无害的寄存器并通过回读确认。

5.4 Corsair RGB 控制器

以下寄存器映射和数据包格式来自 OpenRGB 的 Corsair DRAM 控制器（GPL-2.0）。本文档中的脚本根据该描述重新实现；未复制任何 OpenRGB 代码。在 Hub 处于 I2C 模式时，通过 `0x18` 访问该控制器。

每次访问都是写入 `[register, value]`，或寄存器指针后跟一字节读取。**控制器空闲一段时间后的 第一次访问可能被拒绝一次；请重试。**

表 9. 使用的 RGB 控制器寄存器

寄存器	访问	用途
`0x0B`, `0x21`	写入 `0x00`	复位数据包缓冲区及其 CRC
`0x20`	写入，每次访问一个字节	数据包缓冲区
`0x42`	读取	自复位以来加载的字节的 CRC-8
`0x82`	写入	应用：`0x01` 效果数据包，`0x02` 颜色缓冲区
`0x30`	读取	应用过程中位 3 置位
`0x61` 然后 `0x21`	写入 `0x00`	选择器件信息块
`0x40`	读取，32 次	器件信息块
`0x23`	写入 `0x01`	退出固件更新模式

CRC 为 CRC-8，多项式 `0x07`，初始值 0。应用之前请将其与 `0x42` 比较；不一致表示有字节丢失或重复（重试写入可能导致这种情况），此时不得应用该数据包。

器件信息块。 字节 0-1 为 VID，小端序；字节 2-3 为 PID；字节 28 为协议版本。

固件更新模式。 两端的 LED 都亮红色，是 Corsair 文档中所述的固件更新（引导加载程序） 模式指示。在该状态下，模块报告 PID `0x0B00` 和协议 1，并接受效果数据包（CRC 一致），但 LED 不变化。向 `0x23` 写入一次 `0x01` 后，它变为 PID `0x0B01`、协议 4，出厂彩虹效果随之亮起 （图 1）。模块如何进入该模式尚未确定；它在上电后不久曾被探测过。

> **注意：** 切勿向寄存器 `0x23` 写入 `0x00`。

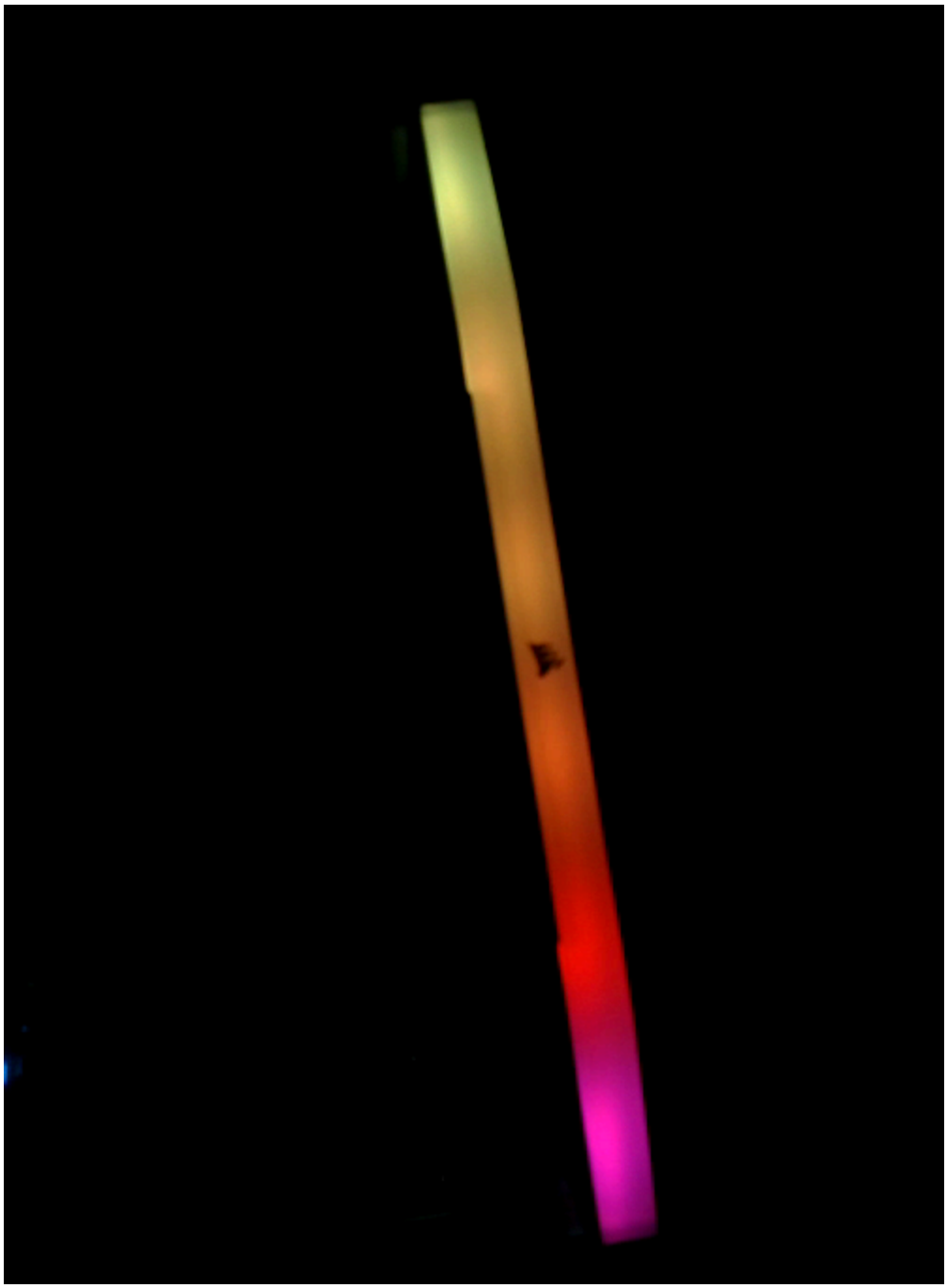


图 1. 退出固件更新模式后的模块：出厂彩虹效果。

颜色缓冲区。 六个 `R, G, B, 0xFF` 条目，每个 LED 一个，共 24 字节，在复位后加载到 `0x20`，检查 CRC，并以 `0x82 = 0x02` 应用。由此可设置单一颜色和逐 LED 颜色（图 2）。该缓冲区 仅在静态模式下显示：在动画效果下，它会被接受（包括 CRC），但会被忽略。因此 `ddr5_rgb.py color` 会先发送静态效果（模式 `0x10`）。

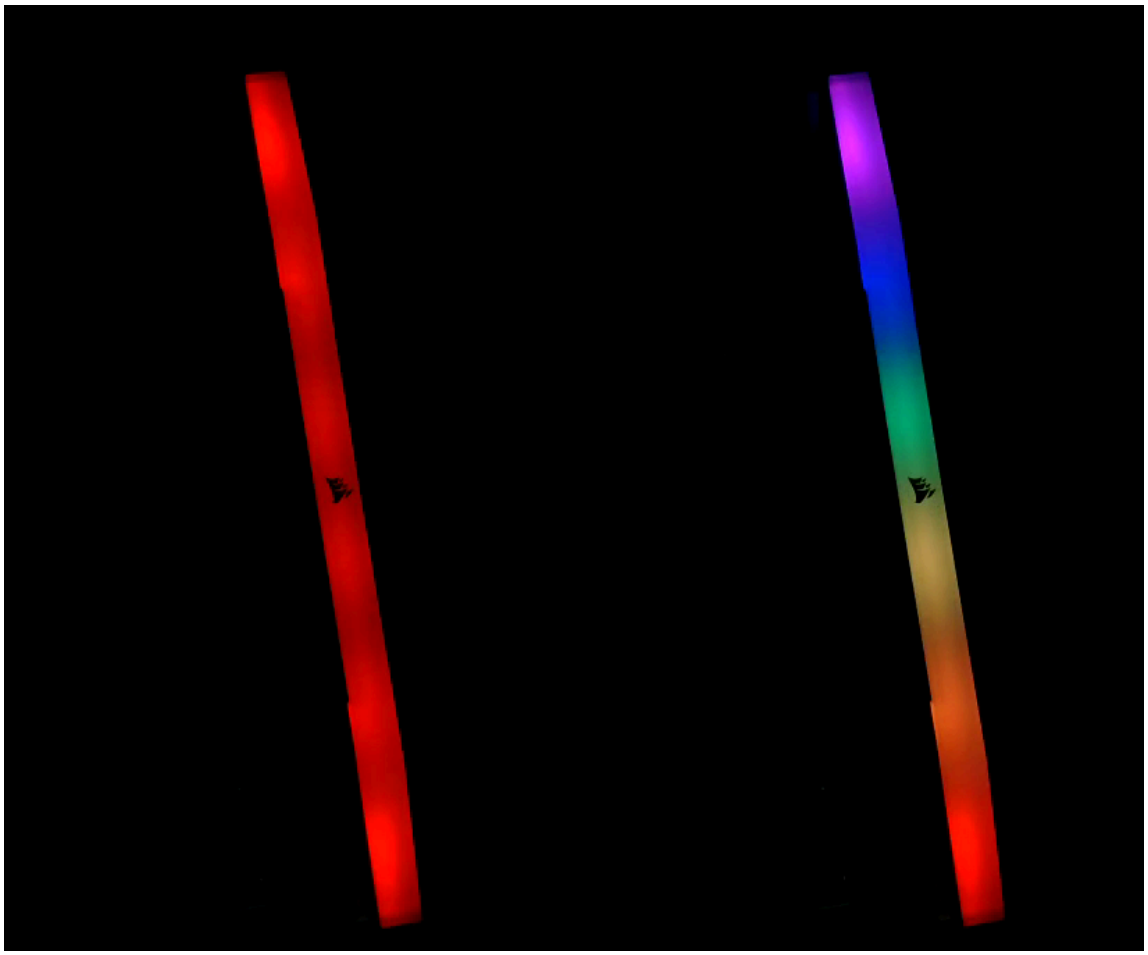


图 2. 颜色缓冲区：六个 LED 全部为红色（左），每个 LED 一种颜色（右）。

效果数据包。 20 字节，以 `0x82` = `0x01` 应用：

```
mode, speed, 0x01, direction, R1, G1, B1, brightness, R2, G2, B2, brightness, 0 x 8
```

表 10. 效果模式

模式	代码	在本模块上观察到
彩虹波（rainbow wave）	`0x03`	是
静态（static）	`0x10`	已应用，但保持了之前的颜色；请改用颜色缓冲区
彩虹（rainbow）	`0x08`	未观察到
脉冲（pulse）	`0x01`	未观察到

6 操作步骤

6.1 上电与连接

通过转接板给模块供电。将电缆连接到 Supernova I3C LV 连接器。

6.2 识别 Hub 并读取其温度

```
python ddr5_spd.py --serial <serial>
```

测试台上的输出：

```
MR0-1 5118  (SPD5118 reads 5118)
MR2 revision 0x14  MR18 0x00  (bit 5 = I3C mode)
MR49-50 CC 01 = 28.75 C
```

6.3 保存 NVM（可选）

将 NVM 保存到您选择的文件中。请将该文件视为可识别该模块的信息：它包含模块的序列号。

```
python ddr5_spd.py --serial <serial> --nvm my_module_spd.bin
```

脚本会恢复 MR11，并报告 SPD CRC 是否一致：

```
MR0-1 5118  (SPD5118 reads 5118)
MR2 revision 0x14  MR18 0x00  (bit 5 = I3C mode)
MR49-50 CC 01 = 28.75 C
NVM: 1024 bytes to my_module_spd.bin, MR11 restored to 0x00
SPD byte 2 0x12 (0x12 = DDR5), byte 3 0x02, CRC 0-509 calc 0x45BF stored 0x45BF OK
```

6.4 以 I3C 模式读取 Hub（可选）

脚本使用 SETAASA 进入 I3C 模式，以 SDR 读取，然后使用 RSTDAA 将 Hub 返回 I2C：

```
python ddr5_spd.py --serial <serial> --i3c --i3c-mhz 1
```

在 I3C 读取期间 MR18 读数为 `0x20`，RSTDAA 之后为 `0x00`：

```
MR0-1 5118  (SPD5118 reads 5118)
MR2 revision 0x14  MR18 0x00  (bit 5 = I3C mode)
MR49-50 CC 01 = 28.75 C
I3C SDR: MR18 0x20  MR0-1 5118  temperature 29.00 C
after RSTDAA (I2C): MR18 0x00
```

在适配器固件 4.6.1 上，每次适配器复位后此操作可执行 11 次；参见第 9 节。

6.5 读取 RGB 控制器的器件信息

```
python ddr5_rgb.py --serial <serial> info
```

测试台上的输出，控制器处于正常模式：

```
VID 0x1B1C PID 0x0B01 protocol 4
```

6.6 退出固件更新模式

如果 PID 为 `0x0B00`（两端为红色），请退出固件更新模式。脚本仅在 PID 为 `0x0B00` 时才写入。

```
python ddr5_rgb.py --serial <serial> leave-bootloader
```

控制器处于固件更新模式时的输出：

```
before: PID 0x0B00 protocol 1
after:  PID 0x0B01 protocol 4
```

以及控制器已处于正常模式时的输出：

```
before: PID 0x0B01 protocol 4
not in firmware-update mode; nothing written
```

6.7 设置颜色或效果

```
python ddr5_rgb.py --serial <serial> color 255,0,0
python ddr5_rgb.py --serial <serial> color 255,0,0 255,80,0 255,220,0 0,255,0
0,0,255 160,0,255
python ddr5_rgb.py --serial <serial> effect wave 255,0,0 0,0,255 --speed 2
python ddr5_rgb.py --serial <serial> demo
```

这些命令成功时不输出任何内容；如果控制器的 CRC 不一致，则以错误退出。

演示序列见 `figures/an0016-video1-rgb-demo.mp4`（42 s），由该脚本所整合自的测试台脚本录制。

7 参考

表 11. ddr5_spd.py 选项

选项	作用
`--serial`	适配器序列号；默认为找到的第一个适配器
`--voltage-mv`	总线电压，默认 1100（LV 连接器）
`--nvm FILE`	通过 MR11 将 1024 字节 NVM 读入 FILE，然后恢复 MR11
`--i3c`	SETAASA，以 SDR 读取 MR18、MR0-1 和温度，RSTDAA
`--i3c-mhz`	`--i3c` 的推挽速率：1、3.125 或 5；默认 1

表 12. ddr5_rgb.py 命令

命令	作用
`info`	检查信息块的 CRC 后，打印 VID、PID 和协议
`leave-bootloader`	如果 PID 为 `0x0B00`，写入 `0x23` = `0x01`；打印前后的 PID
`color R,G,B`	通过颜色缓冲区将六个 LED 设为同一种颜色
带六个 `R,G,B` 的 `color`	每个 LED 一种颜色
`effect MODE C1 [C2]`	效果数据包；MODE 为 static、rainbow、wave 或 pulse；`--speed` 0 到 2
`demo`	红、绿、蓝、白，逐 LED 彩虹，追逐效果，然后彩虹波

两个脚本对每次访问最多重试三次（`ddr5_spd.py`）或五次（`ddr5_rgb.py`）。

8 测量结果

8.1 测试配置

表 13. 测试配置

项目	值
适配器	Binho Supernova，硬件版本 C，固件 4.6.1
总线	I3C LV 连接器，请求 1100 mV，回读 1106 mV
I2C 模式	100 kHz 开漏
I3C 模式	SETAASA，SDR，推挽速率见第 5.2 节
模块	Corsair `CMG16GX5M2E6000Z36`，一条 DIMM，安装在自带 5 V 电源的 DIMM 转接板上
主机	Ubuntu 22.04，pycosmicsdk 1.6.0，Python 3.10
日期	2026 年 10 月 6 日

8.2 结果

表 14. 结果

测量项	结果
Hub 识别，MR0-1	`51 18`
温度，MR49-50	27 到 29 °C
通过 MR11 读取 NVM	1024 字节，字节 0-509 上的 SPD CRC 一致；3/3 逐字节相同
SETAASA，MR18	`0x00` 变为 `0x20`；RSTDAA 后回到 `0x00`
I3C SDR 读取，1 到 10 MHz	1 MHz 下 20/20；5、6.25、7.5 和 10 MHz 下各 50/50
I3C SDR 读取，12.5 MHz	9/50 正确，41 次在成功状态下返回错误数据
I3C 寄存器写入	未按写入的值生效；总线挂起，直到模块重新上电
来自 Hub 的带内中断	已尝试，未收到
在 `0x18` 找到 RGB 控制器	仅限寄存器指针 + 读取
退出固件更新模式	PID `0x0B00` 变为 `0x0B01`，协议 1 变为 4；在两天中共 2 次（一次使用测试台脚本，一次使用 `ddr5_rgb.py`），每次都发现控制器处于固件更新模式
颜色缓冲区，单一颜色和逐 LED	各 10/10，CRC 一致；从正在运行的效果切换到单一颜色，摄像头确认 9/9
效果数据包，彩虹波	10/10

这些是一个模块在一个测试台上的结果，不是规格。随附脚本的计数见第 12 节。尚未进行重新上电后的引导加载程序退出测试。

9 故障排除

表 15. 故障排除

现象	原因	处理
扫描找到 `0x48` 和 `0x50`，但 `0x18` 处没有器件	RGB 控制器不应答裸读取	写入寄存器指针后读取，如 `ddr5_rgb.py info` 所做
对 `0x18` 的第一次访问被拒绝，下一次成功	控制器处于空闲状态	重试；脚本会自动重试
两端 LED 为红色，效果数据包被接受但没有任何变化	控制器处于固件更新模式，PID `0x0B00`	`ddr5_rgb.py leave-bootloader`
之后控制器在未重新上电的情况下又回到固件更新模式	在广播 CCC 流量（RSTDAA/ENTDAA）和适配器复位之后出现；原因未查明	运行 `ddr5_rgb.py info`；如果 PID 为 `0x0B00`，再次执行 `leave-bootloader`
对 Hub 的每次 I2C 读取都返回 `51 FF`，无论寄存器为何	Hub 被之前的 SETAASA 留在 I3C 模式（MR18 位 5 置位）	发送 RSTDAA，或给模块重新上电
多次 I3C 会话后 SETAASA 被拒绝：目标器件表已满（见下文）	适配器固件 4.6.1 在 RSTDAA 之后的每次 SETAASA 都添加一个目标器件表条目；第 12 次失败（3/3 次运行）	复位适配器（`reboot` 或重新插拔）；已在适配器固件中跟踪
颜色被接受（CRC 一致），但 LED 继续播放动画	控制器正在运行动画效果；颜色缓冲区仅在静态模式下显示	`ddr5_rgb.py color` 会先选择静态模式；使用原始写入时，请先发送静态效果，再发送颜色
`0x42` 处 CRC 不一致	加载时有字节丢失或重复	重新发送整个数据包；不要应用
`init_bus()` 找不到 Hub	在本测试台上 Hub 不会被 ENTDAA 发现	使用 `ccc.setaasa([0x50])`
完全没有器件应答	1200 mV 或更高会选择 HV 连接器	请求 1100 mV
I3C 模式下数据错误但状态为成功	推挽速率相对于布线过高	4 MHz 或更低
I3C 写入后每个 I3C 操作都返回 `FW_I3C_ENGINE_BUSY`	Hub 挂起（第 5.3 节）	给模块重新上电；重启 Supernova 无法清除
NVM 转储 CRC 不一致	MR11 被更改，或转储过程中某次读取失败	重新运行；检查 MR11 位 3 为 0

目标器件表已满导致的拒绝报告为 `FW_NOT_ENOUGH_SPACE_IN_TARGET_DEVICES_TABLE`。

10 缩略语

表 16. 缩略语

术语	含义
CCC	通用命令代码（Common Command Code，I3C）
CRC	循环冗余校验（Cyclic redundancy check）
DIMM	双列直插式内存模块（Dual in-line memory module）
ENTDAA	进入动态地址分配（Enter Dynamic Address Assignment，I3C CCC）
HSA	主机边带地址配置引脚（Host sideband address strap）
HSCL, HSDA	主机边带时钟和数据（Host sideband clock and data）
LED	发光二极管（Light-emitting diode）
LV, HV	Supernova 的低电压和高电压 I3C 连接器
MR	SPD5118 的模式寄存器（Mode register）
NVM	非易失性存储器（Non-volatile memory）
PEC	数据包错误校验（Packet error check）
PID, VID	产品 ID，厂商 ID（Product ID, vendor ID）
PMIC	电源管理集成电路（Power management integrated circuit）
RSTDAA	复位动态地址分配（Reset Dynamic Address Assignment，I3C CCC）
SDR	单倍数据速率（Single data rate，I3C）
SETAASA	将所有地址设为静态地址（Set All Addresses to Static Address，I3C CCC）
SPD	串行存在检测（Serial presence detect）
UDIMM	无缓冲 DIMM（Unbuffered DIMM）
VIN_BULK	DIMM 的 5 V 输入电源

11 关于文档中源代码的说明

本文档中展示并在随附归档中提供的示例代码仅供评估和参考。它根据 MIT 许可证发布，许可证文本包含在归档中。

表 17. 资源归档的内容

文件	说明
`AN0016.md`	本文档的 Markdown 源文件
`assets/ddr5_spd.py`	Hub 识别、温度、NVM 读取、I3C 读取检查
`assets/ddr5_rgb.py`	RGB 控制器：器件信息、退出固件更新模式、颜色、效果、演示
`assets/LICENSE`	涵盖示例代码的 MIT 许可证
`figures/an0016-video1-rgb-demo.mp4`	演示序列的 42 s 视频

不提供 SPD 转储：一个转储可识别一个具体的物理模块。

12 修订历史

表 18. 修订历史

修订版	日期	说明
0.1	2026 年 10 月 7 日	初始草稿。第 8 节的首批测量使用了更早的、单独的脚本。随附脚本在同一测试台上运行：`ddr5_spd.py` 在 I2C 模式下 10/10；`--nvm` 3/3，逐字节相同，SPD CRC 一致；`--i3c` 在 1 MHz 下 10/10、在 5 MHz 下 10/10（批次之间复位适配器，见第 9 节），每次 Hub 都回到 I2C 模式；`ddr5_rgb.py` `info` 10/10，`color` 单一颜色 10/10、逐 LED 10/10，`effect wave` 10/10，`demo` 1/1，`leave-bootloader` 1/1（控制器曾有一次处于固件更新模式）。从正在运行的效果发送单一颜色：摄像头确认 9/9 一致。记录的输出：`_shared/tools/documented/ddr5-*.txt`。
1.0	2026 年 11 月 6 日	正式发布。

法律信息（英文）

Definitions

Draft

A document marked draft is under internal review and subject to formal approval. Binho Inc. makes no representation or warranty as to its accuracy or completeness and accepts no liability arising from its use.

Disclaimers

Limited warranty and liability

Information in this document is believed to be accurate and reliable. Binho Inc. makes no representation or warranty, express or implied, as to its accuracy or completeness, accepts no liability for the consequences of its use, and takes no responsibility for content originating outside Binho Inc.

In no event shall Binho Inc. be liable for any indirect, incidental, punitive, special or consequential damages, including without limitation lost profits, lost savings, business interruption, or costs of removing or replacing any product, whether or not based on tort, warranty, breach of contract or any other legal theory. Notwithstanding any damages a customer might incur, the aggregate and cumulative liability of Binho Inc. is limited in accordance with its terms and conditions of commercial sale.

Right to make changes

Binho Inc. reserves the right to change the information in this document, including specifications and product descriptions, at any time and without notice. This document supersedes all information supplied prior to its publication.

Suitability for use

Binho Inc. products are not designed, authorized or warranted for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure of a Binho Inc. product can reasonably be expected to result in personal injury, death, or severe property or environmental damage. Any such inclusion or use is at the customer's own risk and Binho Inc. accepts no liability for it.

Applications

Applications described here are illustrative only. Binho Inc. makes no warranty that they are suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their own applications and products, for appropriate design and operating safeguards, and for determining whether a Binho Inc. product is suitable and fit for their purpose. Binho Inc. accepts no liability for assistance with applications or customer product design.

Third-party products and specifications

This document refers to products, boards, specifications and documentation supplied by third parties, which remain the responsibility of their respective owners. Binho Inc. makes no warranty regarding them and their behavior may change without notice. Register maps, protocol details and device identifiers reproduced here were observed on the specific hardware and firmware versions stated in this document and must be confirmed against the vendor's own documentation for the reader's part.

Example code

Source code supplied with this document is provided as an example, for evaluation and reference. It is not qualified for production use, has not been developed under a functional safety or security process, and is licensed under the terms accompanying it in the distribution archive.

Terms and conditions of commercial sale

Binho Inc. products are sold subject to the general terms and conditions of commercial sale published by Binho Inc., unless otherwise agreed in a valid written individual agreement, in which case only the terms of that agreement apply.

Export control

This document and the items it describes may be subject to export control regulations. Export may require prior authorization from the competent authorities.

Security

All products may be subject to unidentified vulnerabilities or may support security standards with known limitations. The customer is responsible for the design and operation of its applications and products throughout their lifecycle to reduce the effect of such vulnerabilities, and for compliance with all legal, regulatory and security requirements concerning its products.

Trademarks

All referenced brands, product names, service names and trademarks are the property of their respective owners. **Binho** and the Binho logo are trademarks of Binho Inc. **I3C** 和 **MIPI** 是 MIPI Alliance, Inc. 的商标或注册商标。 **JEDEC** 是 JEDEC Solid State Technology Association 的商标。 **Corsair** 和 **Vengeance** 是 CORSAIR MEMORY, Inc. 的商标或注册商标。 **OpenRGB** 是一个开源项目的名称，此处仅用于标识该项目。